J. C. B. Leite* e H. S. Castro**

SUMARIO

O trabalho trata de uma arquitetura de um microcomputador para aplicações em tempo real sob a ótica da tolerância a falhas. Um sistema duplex é proposto aonde a identificação da unidade falha é feita através da comparação de figuras de mérito dos canais. O conceito de dissimilaridade é usado de forma a evitar que falhas em modo comum devido a erros de projeto causem a perda do sistema.

ABSTRACT

This paper investigates the architecture of a microcomputer for real-time applications from a fault-tolerance viewpoint. A duplex system is proposed where the identification of a faulty unit is done by comparison between merit indicators in each channel. The dissimilarity concept is applied with the objective that design errors do not cause a system crash.

* Engenheiro Eletrônico (PUC/RJ, 1974), Mestre em Engenharia Elétrica (PUC/RJ, 1977), Ph.D. (UMIST, 1983); sistemas distribuídos, computação tolerante a falhas; Professor do Depto. de Eng. Elétrica da PUC/RJ, Rua Marquês de São Vicente, 225 - CEP 22453 - Rio de Janeiro, RJ.

** Engenheiro Eletricista (UFCe, 1982), Mestrando em Engenharia Elétrica (PUC/RJ); sistemas distribuídos, computação tolerante a falhas, redes locais; Depto.de Eng. Elétrica da PUC/RJ, Rua Marquês de São Vicente, 225 - CEP 22453 - Rio de Janeiro, RJ.

1. Introdução

O desenvolvimento da indústria de computação e do própio parque industrial brasileiro permite prever que, a médio prazo, técnicas de tolerância a falhas em sistemas de computação venham a ser empregadas em alguns segmentos da indústria nacional.

Esta necessidade já hoje manifesta-se em setores de tecnologia de ponta, como no caso da indústria espacial brasileira. Um exemplo concreto, é o caso do desenvolvimento do computador de bordo do satélite brasileiro, que vem sendo feito no INPE/CNPq, em São José dos Campos.

Como todo e qualquer equipamento, o computador está sujeito a falhas, as quais por sinal devem ser consideradas como elementos normais na vida de qualquer sistema. Como tais, ou seja, como eventos esperados, a elas devemos associar um processo de tratamento que permita a recuperação do sistema e a continuidade de sua operação. Claramente, nem todas as aplicações requerem um alto grau de segurança mas, em certos casos, a interrupção do funcionamento pode ser desastrosa do ponto de vista econômico e em outros pode estar colocando vidas humanas em perigo.

No que se segue, é descrita a implementação da primeira fase do projeto de um microcomputador tolerante a falhas para aplicações em tempo real. O mesmo está em fase de testes devendo em breve o hardware ser declarado operacional. O desenvolvimento de um programa monitor mais elaborado e o estudo e implementação de técnicas de recuperação de erro por programa fazem parte de um outro estudo a ser iniciado.

2. Algumas Definições

Entendemos por um sistema de computação tolerante a falhas aquele que, sem intervenção manual, tem a capacidade de tratar falhas operacionais e de projeto e manter seu comportamento como definido em sua especificação. Duas técnicas complementares são normalmente usadas para a obtenção de sistemas confiáveis [AVIZ 78, DEPL 81]. No primeiro caso, investe-se no uso de melhores componentes, melhor montagem, etc., o que implica em um melhor MTBF. A segunda técnica, faz uso da introdução de redundância para o aumento da confiabilidade. Neste trabalho, estamos mais interessados no segundo aspecto, que envolve maiores considerações conceituais e uso de técnicas específicas para a implementação desta redundância.

Pode-se dividir em quatro fases as estratégias básicas usadas com qualquer tipo de redundância protetiva [ANDE 81]: deteção de erro, avaliação dos danos, recuperação do

erro e tratamento da falha. Em algumas destas fases, mecanismos podem ser implementados para a sua execução, como no caso dos chamados blocos de recuperação. Em outras, devido por exemplo a problemas de custo, simplesmente os problemas são resolvidos a priori. Pode-se citar aqui a fase de avaliação dos danos aonde, por simplificação, as falhas podem ser assumidas ter determinadas características e os danos por elas causados são então analisados e avaliados tão somente na fase de projeto.

Falhas de um modo geral podem ser classificadas em previstas e imprevistas. As primeiras são aquelas que podem ser esperadas com um certo grau de precisão sobre suas causas e consequências. Por exemplo, elas podem ocorrer devido ao envelhecimento de componentes, interferência elétrica, etc.. No segundo grupo estão os modos de falha desconhecidos e interações imprevistas entre o sistema e o seu meio ambiente. Além dessas, ainda no segundo grupo, pode-se acrescentar as falhas de projeto, tanto de hardware quanto de software. Finalmente, um erro é definido como o resultado produzido a partir de entradas incorretas ao circuito em questão causadas pela ação de uma falha.

3.Objetivos do Projeto

Em sua forma mais geral, a meta deste trabalho é o projeto e implementação de um microcomputador de baixo custo, tolerante a falhas, para uso em tempo real. A partir daí, uma série de pontos podem ser levantados, que são a seguir detalhados. Exemplos de sistemas deste tipo podem ser encontrados em [KAME 80, PATE 82 , TOY 78].

Primeiramente por motivos de custo e complexidade, optou-se pela realização de uma estrutura em **dois canais**, em detrimento de estruturas TMR ou outras mais elaboradas. Neste caso, o mecanismo básico de deteção de erros consiste na comparação das saídas dos dois canais paralelos, que operam sobre o mesmo conjunto de dados de entrada. É importante notar que a localização de uma falha fica mais difícil, embora esta ainda possa ser realizada com o suporte da análise dos erros e de testes. Se os canais utilizados são indênticos, o que facilitaria a elaboração do software de suporte, problemas surgiriam ao nível das falhas não antecipadas (p.ex., falhas de projeto), que propagar-se-iam da mesma maneira em ambos os canais, impedindo então sua deteção. Desta forma, optou-se pelo uso da chamada redundância dissimilar, que pode fornecer um mecanismo de deteção de erros bem eficiente [MESQ 79, RIAL 80]. Obviamente isto é um fator de complicação , e para minorar o seu impacto ao nível da complexidade, os canais foram projetados com base nos microprocessadores Z80 e 8085. Como é sabido, o conjunto de instruções do 8085 é um subconjunto do conjunto das instruções do Z80, o que pode simplificar a fase inicial de depuração do sistema. É importante ressaltar que cuidados devem ser tomados para que a implantação do software de suporte em cada canal seja feita

com diferentes codificações, e possivelmente diferentes algoritmos. Esta última decisão deve ser pesada em relação ao aumento em custo a ser incorrido.

Outro ponto importante é a exigência de **não existir uma unidade central de gerenciamento**, ou ainda que, no caso da existência de elementos críticos ('hardcore'), que estes tenham confiabilidade bem mais elevada do que um canal básico. Um exemplo de tal elemento crítico é o chamado seletor de saída. Em um sistema duplex, o elemento que faz a comparação entre as saídas dos canais é, essencialmente, uma unidade de gerenciamento. Assim, sua operação deve ser livre de erros, pois uma falha na comparação pode permitir a propagação de uma saída errônea, o que vai contra os objetivos do projeto. De modo a reduzir-se o riscos de falhas e ainda, para evitar a existência de elementos centrais, decidiu-se realizar a comparação por software, pela troca de informações entre os canais. Além disso, os própios dados enviados ao seletor de saída, bem como a saída do própio seletor devem ser realimentadas. Vale observar que o procedimento de comparação por software é mais lento do que caso o mesmo fosse implementado em hardware. Isto no entanto não constitue um problema nesta máquina experimental. Finalmente, uma restrição fundamental é que não haja um elemento central de sincronização.

Um fato importante para um microcomputador tolerante a falhas é que o mesmo sempre forneça saída correta, a menos que haja uma indicação de erro. Um sistema com tal característica é chamado de **seguro a falhas.** Assim, além das falhas simples ele deve também ser capaz de detetar ou evitar as chamadas falhas em modo comum. Isto é aqui conseguido pelo uso de canais dissimilares e independentes. Além disso, o software de comunicação entre os canais deve ser livre de bloqueio e um canal não pode ter a permissão de interromper o outro.

Outra característica exigida neste projeto é que a estrutura tenha **capacidade de degradação.** Ou seja, o microcomputador deve ser capaz de detetar falhas permanentes e transientes, tentar sobreviver aos seus efeitos e, caso não seja possível a recuperação do canal atingido, funcionar em modo simplex, gerando um alarme relativo a esta situação. Para que esta capacidade de degradação possa ser implementada é necessário que mecanismos de deteção de erro sejam implantados e que se faça a indicação do canal falho. Os mecanismos de deteção aqui usados são:

- 'time-out' em intercomunicações;
- paridade na memória;
- comparação de entradas, saídas e trocas de mensagem;
- endereçamento fora de limites;
- programas de testes periódicos;
- 'rollback' em programas repetitíveis.

Além disso, estão previstos testes de validade sobre os limites dos valores de entrada e saída, sua taxa de variação, etc.. Com isto, outras situações anormais não detetadas pelos mecanismos referidos podem ser indicadas.

Finalmente, toda redundância empregada deve ser **transparente** ao usuário. Mecanismos devem ser elaborados tais que a implementação da tolerância a falhas possa se dar automáticamente. Assim, além dos diversos detetores, inserção de pontos de recuperação, etc., técnicas como as propostas em [LOQU 85] devem ser investigadas. Isto contudo faz parte de um outro estudo, relativo a implementação do monitor de controle.

4.Sincronização

Em virtude das diferentes caracterfsticas dos processadores usados e com o objetivo de eliminar elementos centrais, os canais desta máquina funcionam de forma independente, cada qual com sua própia base de tempo, sincronizando-se via troca de mensagens.

Duas formas de sincronização estão previstas, uma associada ao infcio de ciclo de operação e outra quando da troca de informações entre tarefas. Assim, admite-se a existência de um ciclo de operações em tempo real ao infcio do qual os canais trocam informações e reiniciam suas bases de tempo. Isto é necessário pois como são previstos canais distintos, com requerimentos próprios de relógio, fatalmente há a propagação de uma (pequena) diferença entre as bases de tempo dos canais. Além disso, é também fornecido o suporte para a troca de informações entre tarefas durante um ciclo de operações. O diagrama da figura 1 dá uma indicação do mecanismo de sincronização.

5.Descrição do Sistema

O sistema foi projetado em torno de uma estrutura baseada em dois canais fazendo-se uso do conceito de redundância dissimilar. A sua arquitetura está indicada na figura 2. Cada um dos canais é um microcomputador capaz de, isoladamente, assumir o processo de controle. Assim, ambos terão estrutura semelhante, baseada em uma arquitetura clássica, contendo os mesmos tipos de recursos. Do ponto de vista da implementação, o sistema foi realizado em três placas, sendo uma para cada canal e a terceira para a implementação do seletor de saída e parte da comunicação com o sistema de desenvolvimento MCPUC-80.

Cada canal contém os recursos abaixo relacionados, sendo alguns deles associados ao processo de deteção e recuperação de falhas:

- unidade central de processamento de 8 bits;
- 4k de EPROM;

- 4k x 9 de memória RAM (8 bits mais paridade) e armazenamento de endereço errôneo;
- registros para limites de endereço de execução;
- unidade de comunicação inter-canal;
- unidade de temporização;
- unidade de E/S;
- unidade de comunicação serial com o sistema de desenvolvimento;
- indicadores de status.

O teste de cada parte do sistema é feito ou através de indicadores próprios, como no caso da memória RAM, ou através de temporização, como no caso das comunicações entre canais, ou por comparação, como no caso das saídas ou ainda por testes periódicos. Algumas unidades podem ter mais de um tipo de teste de funcionamento. É importante ressaltar que da eficiência destes testes dependerá o sucesso da operação de deteção e recuperação, ou seja, da **cobertura** desta operação. Este ponto será abordado quando da avaliação da confiabilidade.

Para exame da memória ROM ficaríamos limitados a operações de sequências de teste periódicas, o que pode não ser conveniente dado que a deteção do erro poderia ficar muito distante no tempo do instante de sua ocorrência. Assim, optou-se por um esquema aonde, quando da gravação da ROM é acrescentada uma última palavra de teste sobre todo o conteúdo anterior. Quando o sistema é ligado, o conteúdo desta ROM é transferido para a RAM, a transferência testada para detetar a ocorrência de erros, dando-se então a execução a partir do programa gravado agora em RAM, que é eficientemente testada pelo esquema de paridade.

Quanto ao seletor de saída é implementado usando-se um esquema de multiplexação aonde a seleção de quem comanda a saída é feita independentemente por cada canal, através de uma porta ou-exclusivo. Assim, mesmo que o comando de saída de um canal 'cole' em um certo valor, ainda é possível fazer a escolha correta da saída a partir do outro canal. Esta parte do circuito foi deliberadamente mantida pequena para garantir que sua confiabilidade seja alta quando comparada com a do sistema que a precede. É interessante lembrar que esta parte é um elemento série do ponto de vista de confiabilidade.

Finalmente vale acrescentar que as comunicações inter-canal são redundantes, sendo todas as trocas de mensagem limitadas em tempo. Um protocolo de comunicação pode ser eficientemente implementado fazendo-se uso de portas de E/S paralelas especiais, como por exemplo a Intel 8255.

6.Estimativa da Confiabilidade

O uso de redundância só se justifica no caso aonde a mesma é capaz de fornecer algum grau de proteção. Desta forma, esta proteção deve ser quantificada para haver a garantia de que a mesma é efetiva nas condiçoes de operação e dentro do chamado tempo de missão.

Vários parâmetros podem ser escolhidos como indicadores desta proteção, sua escolha sendo dependente do uso a que se destina o sistema em questão. Neste caso, estamos interessados em um sistema que funcione ininterruptamente por um determinado período de tempo, que não excede 100 horas. Assim, um parâmetro adequado para a quantificação da proteção oferecida pela redundância é a confiabilidade. Esta é definida como a probabilidade de que o sistema funcione corretamente, nas condições especificadas, por um dado período de tempo T, referido como tempo de missão.

Uma das formas mais aceitas e usadas para a estimativa da confiabilidade é a chamada modelagem analítica, da qual faremos uso. É assumido aqui que o processo de chegada das falhas é caracterizado pela lei Poisson e que os tempos associados a ação de recuperação são exponencialmente distribuídos.

Seja então o modelo da figura 3 aonde os números nos estados indicam a quantidade de unidades boas, c é a cobertura da ação de recuperação e os σ_i representam taxas de falha. Aplicando os métodos clássicos de solução para sistemas Markovianos, chegamos ao resultado abaixo:

$$R(t) = p_2(t) + p_1(t)$$

$$p_2(t) = e^{-\sigma_2 t}$$

$$p_1(t) = \begin{cases} c\sigma_2 t e^{-\sigma_1 t} & \text{para } \sigma_1 = \sigma_2 \\ \dfrac{c\sigma_2}{\sigma_2 - \sigma_1}\left[e^{-\sigma_1 t} - e^{-\sigma_2 t}\right] & \text{para } \sigma_i \neq \sigma_2 \end{cases}$$

Para a ação de recuperação podemos usar um modelo baseado em CAST [CONN 77, GEIS 83] (figura 4) aonde **d** é a probabilidade de deteção da falha, **f** é a probabilidade de que haja fuga de uma falha transiente (ou seja, a mesma é tomada como permanente), **e** é a

eficiência da ação de recuperação de falhas permanentes, λ é a taxa de falhas permanentes e μ é a taxa de falhas transientes.

Dado que os tempos envolvidos nos estados DET, TR e PM são muito menores do que os tempos associados aos outros estados, o modelo da figura 4 pode ser reduzido ao da figura 3, aonde σ_2, σ_1 e c são dados por:

$$\sigma_2 = 2(1 - d)(\lambda + \mu) + 2d(\lambda + f\mu)$$

$$\sigma_1 = (\lambda + \mu)$$

$$c = \frac{2de(\lambda + f\mu)}{\sigma_2}$$

Como pode-se notar os σ_i representam as taxas de falha efetivas e c, a cobertura, é a probabilidade de chaveamento correto.

Além dos dois canais, para que o sistema possa efetivamente operar, tem-se o seletor de saída, que neste caso é um elemento série do ponto de vista de confiabilidade. Desta forma a confiabilidade do sistema é dada pelo produto da confiabilidade do seletor pela a confiabilidade da estrutura paralela.

Ter-se-ia ainda de considerar a fonte de alimentação, mas como o objetivo aqui é observar os ganhos relativos a uma unidade simplex, que também deve ser alimentada, isto não é feito. Assim, os gráficos que se seguem representam tão somente a confiabilidade da estrutura duplex comparada com a de uma estrutura simplex.

Na figura 5 temos um gráfico comparativo entre as estruturas duplex e simplex para o caso aonde a cobertura é de aproximadamente 90%, que pode ser considerado um valor baixo [RENN 84]. As curvas {1,2} e {3,4} representam limites para a confiabilidade das estruturas duplex e simplex, respectivamente, para as variações possíveis na taxa de falhas de um canal. O efeito do seletor de saída na estrutura duplex foi considerado, embora o mesmo seja bem pequeno. Note-se ainda que no caso simplex não foram consideradas as falhas transientes, responsáveis pela maioria das interrupções em um sistema.

7.Conclusão

O sistema apresentado representa parte do esforço desenvolvido no Grupo de Sistemas de Computação do DEE-PUC/RJ na área de computação tolerante a falhas. Certamente o mesmo não é um produto acabado, muito ainda tendo de ser feito. Podemos citar aqui a gerência de reconfiguração e o tratamento de eventos assíncronos. Os autores gostariam de agradecer aos Profs. O. G. Loques Filho e G. F. G. da Silveira pela leitura do texto e suas sugestões de melhoria. Contudo, quaisquer incorreções são única responsabilidade dos autores.

8.Referências

[ANDE 81] T.Anderson e P.A.Lee, "Fault-Tolerance: Principles and Practice", Prentice/Hall International Inc., Londres, 1981.

[AVIZ 78] A.Avizienis, "Fault-Tolerance: The Survival Attribute of Digital Systems", Proc. of IEEE, Vol.66, No.10, Outubro de 1978, p.1109-1125.

[CONN 77] R.Conn et al., "CAST - A Complementary Analytic-Simulative Technique for Modeling Complex Fault-Tolerant Computing Systems", Proc. AIAA Comput. Aerospace Conf., Los Angeles, C.A., Novembro de 1977.

[DEPL 81] P.G.Depledge, "Fault-Tolerant Computer Systems", IEE Proc., Vol.128, Part-A, No.4, Maio de 1981, p.257-272.

[GEIS 83] R.M.Geist e K.S.Trivedi, "Ultrahigh Reliability Prediction for Fault-Tolerant Computer Systems", IEEE Trans. on Computers, Vol.C-32, No.12, Dezembro de 1983, p.1118-1127.

[KAME 80] M.Kameyama e T.Higuchi, "Design of Dependent-Failure-Tolerant Microcomputer System Using Triple-Modular Redundancy", IEEE Trans. on Computers, Vol.C-29, No.2, Fevereiro de 1980, p.202-206.

[LOQU 85] O.G.Loques Filho, "Uma Arquitetura Flexível para Sistemas Distribuídos Tolerantes a Falhas", submetido ao SEMISH 1985.

[MESQ 79] A.C.Mesquita Jr., "Tolerância a Falhas Transientes por Diversificação", Tese de Mestrado, Dep. de Eng. Elétrica, PUC/RJ, Agôsto de 1979.

[PATE 82] D.C.Patel, "Dual Dissimilar Processor System for High Reliability", Ph.D. Thesis, Dept. of Electrical Eng. and Electronics, University of Manchester Institute of Science and Technology, Outubro de 1982.

[RENN 84] D.A.Rennels, "Fault-Tolerant Computing - Concepts and Examples", IEEE Trans. on Computers, Vol.C-33, No.12, Dezembro de 1984, p.1116-1129.

[RIAL 80] A.Rial F., "Dissimilar Redundancy in Fault-Tolerant Microcomputer Systems", Ph.D. Thesis, Dept. of Electrical Eng. and Electronics, University of Manchester Institute of Science and Technology, Outubro de 1980.

[TOY 78] W.N.Toy, "Fault-Tolerant Design of Local ESS Processors", Proc. IEEE, Vol.66, No.10, Outubro de 1978, p.1126-1145.

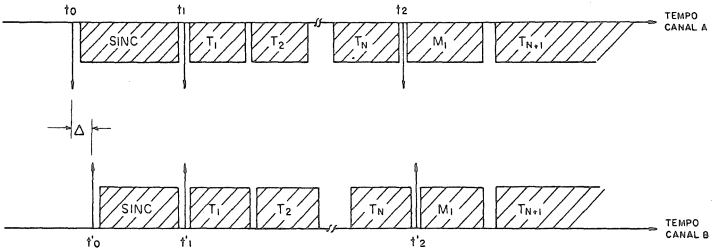


Figura 1 - Sincronização

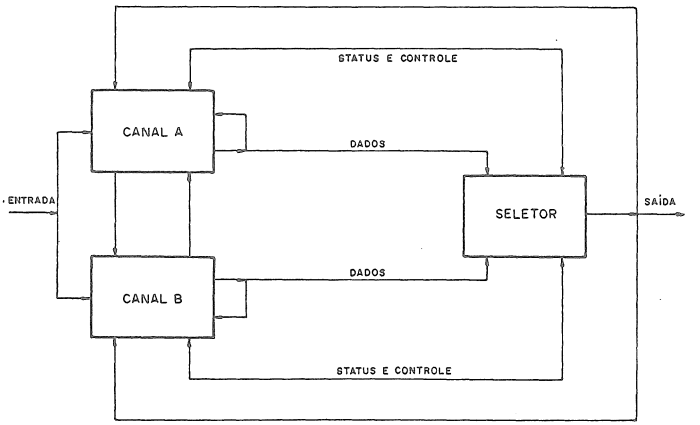


Figura 2 - Arquitetura Básica

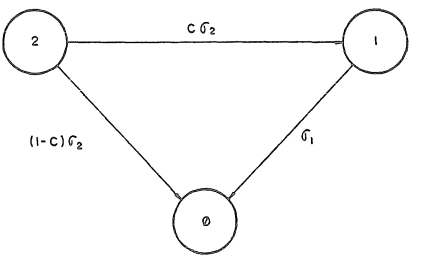


Figura 3 - Modelo de Falhas

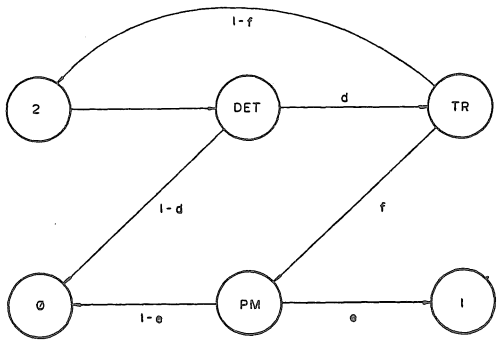
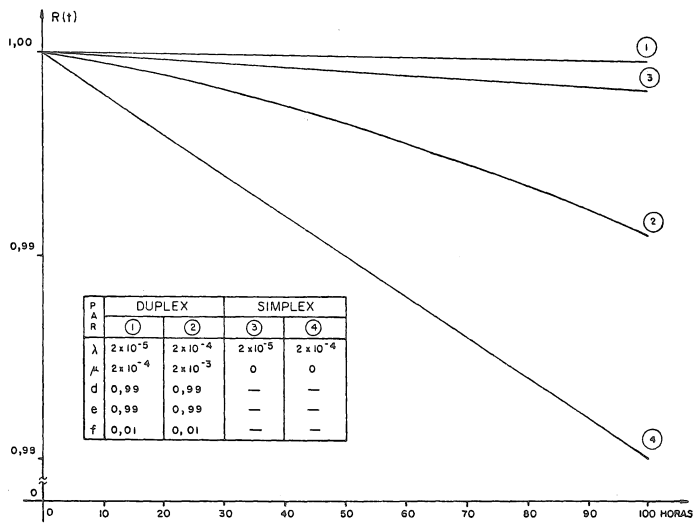


Figura 4 - Modelo de Cobertura



PAR	DUPLEX		SIMPLEX	
	①	②	③	④
λ	2×10^{-5}	2×10^{-4}	2×10^{-5}	2×10^{-4}
μ	2×10^{-4}	2×10^{-3}	0	0
d	0,99	0,99	—	—
e	0,99	0,99	—	—
f	0,01	0,01	—	—

Figura 5 - Confiabilidade: Duplex x Simplex